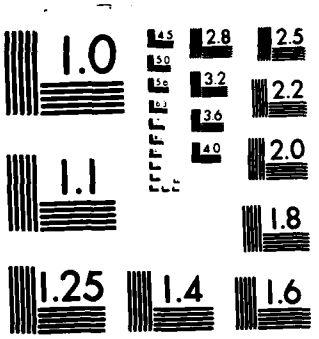
AD-A146 455 ONR LONDON WORKSHOP ON COMPUTER ARCHITECTURE 16-17 MAY 1/1
1984(U) OFFICE OF NAVAL RESEARCH LONDON (ENGLAND)
J F BLACKBURN 19 JUL 84 ONRL-C-2-84

UNCLASSIFIED F/G 9/2 NL

END
DATE FILMED
11-84
DTIC



(12)

ONR LONDON REPORT

C-2-84

OFFICE OF NAVAL RESEARCH

BRANCH OFFICE LONDON ENGLAND

ONR, LONDON, WORKSHOP ON COMPUTER ARCHITECTURE

J.F. BLACKBURN

19 JULY 1984

UNITED STATES OF AMERICA

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER C-2-84	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ONR, London, Workshop on Computer Architecture		5. TYPE OF REPORT & PERIOD COVERED Conference 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J.F. Blackburn		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Office of Naval Research Branch Office London Box 39 FPO NY 09510		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 19 July 1984 13. NUMBER OF PAGES 10
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer architecture
Hardware design
Parallel processing systems

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution
Availability Codes
Dist | Avail and/or Special
A-1

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The US Office of Naval Research, London, hosted a workshop on computer architecture on 16 and 17 May 1984. The meeting brought together 12 of the leading research professionals in European universities and several interested persons from the US government to discuss the present state of research in this important field and to assess future directions. Most of the discussion was on hardware design; one presentation concentrated on the programming of parallel processing systems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ONR, LONDON, WORKSHOP ON COMPUTER ARCHITECTURE

The US Office of Naval Research, London, hosted a workshop on computer architecture on 16 and 17 May 1984. The meeting brought together 12 of the leading research professionals in European universities and several interested persons from the US government to discuss the present state of research in this important field and to assess future directions. Most of the discussion was on hardware design, but one speaker, Professor Schendel, concentrated on the programming of parallel processing systems. A list of participants is given in Appendix A.

Trends in Computer Architecture

The first speaker, Dr. Paul Schneck, discussed US trends in computer architecture and their impact on scientific computing. He began by briefly describing vector processors that have been designed and built in the US:

1. The ILLIAC IV uses emitter coupled logic (ECL), which is very fast and requires users to design with the higher speed in mind and to follow certain layout rules. ECL requires only a 1-V swing in 3 to 4 nanoseconds. Cable tuning in the ILLIAC IV was to a fraction of a centimeter. It is a single-instruction, multiple-data (SIMD) machine with 64 processors.

2. The Control Data Corporation (CDC) STAR was a vector-processor forerunner to the CRAY-1, a scalar processor with vector component. It is mainly used as a scalar processor.

3. The CRAY-XMP is a two-processor repackaging of the CRAY-1. A four-processor version was to be announced in the summer of 1984.

4. The ETA is a follow-on to the CDC 203. It uses low-power, mid-speed components with high density.

5. The Denelcor HEP-1 (Heterogeneous Element Processor) is a hardware multiprogram medium system using conventional transistor-transistor logic (TTL) packaging.

The US Navy is interested in high-speed computation useful for simulation, which is faster and easier than experimentation. For example, Airbus in the aerospace industry used simulation to design its aircraft fuselage. The Boeing Company used simulation to determine engine placement in the Boeing 737 to minimize drag. The Atlantic Richfield Company achieved a payback on its CRAY-1 computer in 6 months of use.

Since 10^9 floating point operations per second appear to be the limit for a single processor, there is a need for concurrency in system design. Up to 16 concurrent processing systems are available or planned in the commercial marketplace.

There are at present 70 projects in the US on large-scale parallelism. The US National Aeronautics and Space Administration (NASA) is sponsoring a massively parallel processing system being built by Goodyear Aerospace. It consists of a 128×128 array of 1-bit processors at 10 MHz. Performance of 160 billion instructions per second is expected. It will be a picture processor used for meteorological modeling. It is a SIMD machine with eight processors per chip and using complementary metal-oxide-semiconductor technology.

A project at New York University intends to achieve a parallel system with 4000 processors. And the Massachusetts Institute of Technology is developing a data-flow parallel processor. Other systems include Connection Machine by Thinking Machines Co., COSMIC Cube by the California Institute of Technology, DARPA by Strategic Computing, and CEDAR by the University of Illinois.

Data Structure Architecture

Professor Wolfgang Giloi gave a brief description of the STARLET computer, developed at the Technical University of Berlin, as an example of data-structure architecture. Such systems manipulate at the hardware level arbitrarily complex data structure objects as entities. Only the entire data structure is referenced by name, and substructures or single data items of a

data-structure object are accessed by the execution of access functions. A data-structure machine is equipped with very fast, dedicated access processors that execute the access functions. The conceptual bottleneck, present in conventional systems, is avoided because the computer supports procedural or functional programming languages that allow the state of arbitrarily complex data-structure objects to be transformed by one complex operation, invoked by a single instruction of function application. The physical bottleneck does not exist since data items are moved to and from memory not by execution of move instructions, which would have to be fetched from memory, but by use of an address stream calculated by the access processor at high speed. Thus the inherent parallelism of data-structure objects is exploited for parallel processing in the SIMD made.

Giloi also mentioned specialized systems derived from the STARLET architecture.

The Development of Algorithms for Parallel Processing Systems

As Udo Schendel said in his introduction, numerical algorithms can be classified in different ways--for example, as algebraic or analytic algorithms; as finite or infinite algorithms; or as direct or iterative algorithms. In recent years a new classification has become more important: the difference between serial and parallel algorithms, as a result of the development of parallel and pipeline computers. These machines allow the parallel execution of arithmetic operations and are able to handle large amounts of information. The basic idea of a parallel computer is that programs which use n processors are generally n times as fast as programs using one processor. However, experience and theory show that the speed-up is really smaller.

Thus, there is common interest in a numerical process that makes optimal use of parallel computers. Another theoretical question is how to solve problems using maximum parallelism.

A Data-Flow System--Design and Performance

Data-flow systems are based on the execution of two-dimensional graphical machine code in which instructions that are available for concurrent execution are written alongside one another, while those that must be executed in sequence are positioned one under the other. Data dependencies between individual instructions are indicated by directed arcs linking them together. Instructions do not reference memory, since the data-dependence arcs allow data to be transmitted directly from generating instruction to subsequent instruction. Thus instructions can be viewed as pure operations, and each instruction can be activated independently by incoming data values. Execution begins as soon as all required input values for that instruction have arrived.

Data-flow systems differ in the way they handle re-entrant code. Static systems do not permit concurrent re-entry and are restricted to implement loops rather than recursion. Dynamic systems permit recursive re-entry, either by code copying or by token tagging at the outset of each recursive activation. The nature of a system determines the type of language feature that can be supported. Recursion cannot be handled by static systems.

John Gurd described the data-flow system developed at the Victoria University of Manchester, UK. The basic structure of the Manchester system incorporates a processing unit, a token queue, a matching unit, a node store, and a host computer to provide peripheral control and storage. The units are connected in a pipelined ring around which the tokens flow. The tokens carry data, a label, and a destination node address. A token produced by a node exits from the processing unit, where the execution of several node operations may be proceeding concurrently. Upon arrival at the token queue the token is stored temporarily. The matching unit collects pairs of tokens with the same destination node address and label. If no partner is found, the token is

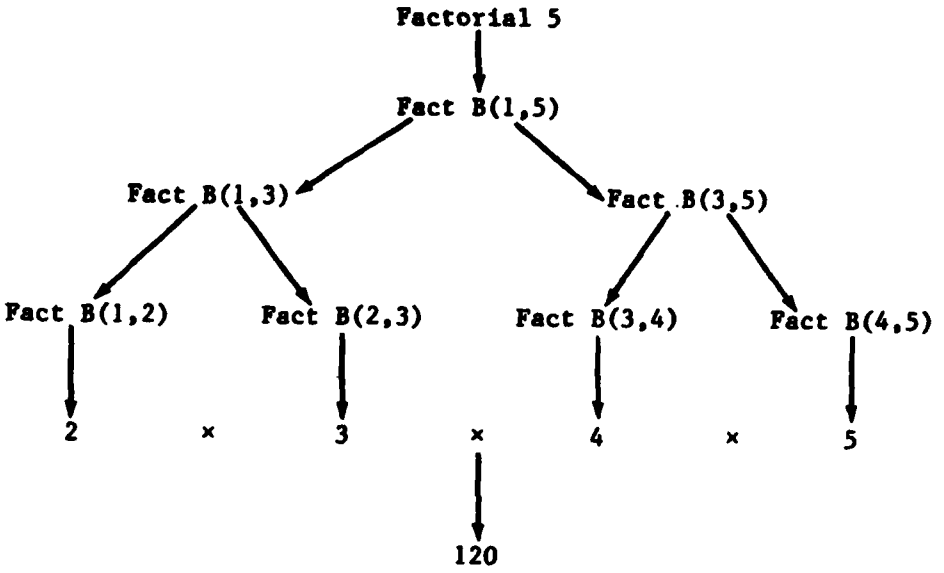


Figure 1. Computation of factorial 5.

written in the store to await a partner. A pair of matched tokens leaving the matching unit addresses the node store to obtain the destination node operation and the subsequent destinations of its outputs. The package is then passed to the processing unit for execution.

Preliminary evaluation of the 20-processor system at Manchester has shown that a wide variety of quite small programs contain sufficient parallelism to exhibit impressive speed-up versus the number of active function units in a single-ring system. More work is needed to determine the behaviors of large programs which cause matching store overflow.

A Graph Reduction Processing System

Scientists at Imperial College, London, are developing a graph reduction system that uses the transputer, a single chip microprocessor made by INMOS Ltd., as the basic component. The idea is to have large numbers of parallel activities in the system. A simple example of parallel reduction is that of computing factorial n:

$$[n! = n(n-1) \cdot \cdot \cdot 2 \cdot 1].$$

The following program consists of sets of equations defining functions:

Factorial: Integer ⟶ Integer

Factorial n = Fact B (1,n)

Fact B: Integer
× Integer ⟶ Integer

Fact B (i,i) = 1

Fact B (i,i+1) = i+1

Fact B(i,j) = Fact B(i,mid)
× Fact B(mid,j), where
mid = Integer Divide (i+j,2)

Figure 1 illustrates the computation of factorial 5.

In the Imperial College system, the graph of an expression is represented by a collection of packets, each of which represents one node of the graph, the arcs extending down from that node, and necessary control information. A packet consists of three primary and three secondary fields. The former contain the information required to represent a node, while the latter contain the control information required for evaluation. The functions of the three primary fields are as follows:

1. Identifier--contains an identifier unique to the packet and provides a name by which the packet may be referenced.

2. Function--contains the function associated with the node represented by the packet.

3. Argument List--contains the identification of the packets representing the offspring of the node (i.e., the arguments of the function). When the packet is used to represent a numeric value, the argument-list field is replaced by the value field that contains the binary representation of the value.

A 20-processor system is expected to be operational by the summer of 1985.

Von Neumann Structures as Basic Cells of New Computer Architectures--Compiling Them in Silicon

Professor F. Anceau asserted that the best implementations of non-Von Neumann architectures are made with single or cluster Von Neumann-processors blocks. He gave examples of pipeline computers, data-flow computers, LISP computers, and systolic arrays.

Technology provides memory elements as capacitances or flip flops and computing elements, with no mixing between memory and computing. Multiplexing of a few computing elements for all computations is accomplished in the Von Neumann architecture.

Anceau discussed component levels--including transistors, gates, macro-logic, processing components, computer components, system components, and machine components. Each component is built with an interconnection of a set of components of a lower level. He defined a Von Neumann computer component as a device which executes an algorithm. Computer-like structures are good engineering solutions for many special-purpose integrated circuits.

The Von Neumann architecture gives the best adaptation to hardware possibilities. Computer-like blocks are the cells of many very large integrated circuits. A large subset of custom-oriented integrated circuits may use a Von Neumann computer-like internal architecture.

Large-Systems Architecture at the University of Lille

At the University of Lille, France, research in parallel processing includes the development of communication tools and multiprocessors. V. Cordonnier discussed data-driven architectures; operating systems, including a distributed system driven by capabilities; and image processing. Considerable emphasis is given to image processing at Lille. Cordonnier discussed algorithm-oriented architecture, pixel-oriented architecture, and object-oriented architecture. A very fast machine called MAP has been developed for image processing with effective real-time capabilities. It contains 16 processing elements and 16 memory banks with a switching network allowing access of any processor to any memory bank.

Cordonnier described the following limits of data-driven architecture: lack of dynamic control, very high data throughput, the need for unusual architecture in processing elements, and the need for some associative facilities.

Both the static and the dynamic versions of data-flow architecture were discussed.

Finally he mentioned a new project with the following goals:

1. To increase the number of processing elements to more than 100;
2. To suppress the associative access;
3. To suppress the circulating memory;
4. To separate control blocks from execution blocks;
5. To increase the power and the responsibilities of the compiler.

Parallel Algorithms in Conventional Linear Algebra

Professor D.J. Evans presented techniques for exposing parallelism in a problem. Several new parallel algorithms for the direct and iterative solution of linear systems were presented and compared with existing sequential

methods. A new explicit method for the finite difference solution of parabolic differential equations was derived. This new method uses stable asymmetric approximations to the partial differential equations which, when coupled in groups of two adjacent points (four points for two dimensions) on the grid, result in implicit equations that can be easily converted to explicit form. They offer many advantages--especially for use on parallel computers. The judicious use of alternating this strategy on the grid points of the domain results in new explicit parallel algorithms which possess unconditional stability.

Systems Architecture Research at Cambridge University

Professor D.J. Wheeler gave a brief summary of Cambridge University's long history of research in computers and computing. Recent work includes the Cambridge ring, which is a well-established local-area network system throughout the UK. Much of the work at Cambridge now is on local-area network research.

The Cambridge fast ring is a high-speed local-area network similar to the Cambridge ring but much faster. In addition to allowing much higher transfer rates than existing networks, it incorporates facilities for partitioning the bandwidth between several groups of users and for connecting rings together without the need for bridge computers and complex addressing schemes. For the fast Cambridge ring, a raw data rate of 100 MB/s was chosen as a target, and the design was required to be realizable in very large scale integration (VLSI). Also, fiber-optic connections were to be usable. The slotted ring was chosen as a suitable framework for the design of the fast network.

A prototype computer called SKIM was built at Cambridge in 1982-83. It was constructed on four wire-wrapped, extended, double Eurocards; a micro-code random access memory (RAM) card; a central processing unit card; a RAM controller card; and a RAM array card. Up to three more RAM array cards could be added.

All input/output from the processor is done via a host computer system. The host and SKIM together form a stand-alone working system. There is a link to the local main frame, where most of the system software is being developed.

TTL has been used throughout the system. The only non-TTL devices are the memory chips. All four boards contain a total of 230 chips, of which 92 are memory chips. Further details on this computer are contained in Technical Report No. 40, University of Cambridge, Computer Laboratory, February 1984.

Parallel Processing Architectures

In his second talk Giloi discussed strongly coupled and loosely coupled systems and expressed his preference for the latter. He defined a strongly coupled parallel processing system as one consisting of one or more processing sites, all of which have access to a shared control memory that accommodates the data objects to be manipulated.

A loosely coupled system consists of a number of processing nodes, which communicate with each other by exchanging messages. A processing node is a processor with private memory and, if needed, private peripherals.

The performance of a strongly coupled system is limited by the maximum obtainable memory bandwidth. The performance of a loosely coupled system is limited by the maximum obtainable internode communication bandwidth. However, the decision as to whether a system should be strongly or loosely coupled depends also on the nature of the algorithms to be performed.

Loosely coupled systems are adequate when functions (processes) exhibit strict locality, whereas strongly coupled algorithms cannot be mapped very well onto loosely coupled systems.

Hierarchical Distributed Computer Architectures

Professor G.H. Granlund described a special computer developed at Linköping University for processing structural

information--e.g., image processing. The general operator processor (GOP) is a multiple instruction-multiple data (MIMD) system using parallelism and pipelining extensively. For the special applications for which it was designed, the system is reported to be 100,000 times faster than nonparallel systems using more conventional techniques.

Image information is structural: the information is provided partly by the data values in a space and partly by the structural relationships between these data values. This implies difficulties in processing by conventional computers, where knowledge of class membership of data is implicitly assumed. The GOP computer provides adaptivity to the data by doing the processing in a hierarchical structure.

A minicomputer version of the GOP computer will be marketed by Context Vision, a Swedish company, and will be available by the end of 1984.

Characterizing Parallel Computer Architectures

Professor Roger Hockney gave a two-parameter description of a computer which characterizes the performance of serial, pipeline, and array-like architectures. The first parameter is the maximum performance in millions of floating point instructions per second. Another parameter measures the apparent parallelism of the computer. He stated that for computers with a single instruction stream, the relative performance in processing two different algorithms on the same computer depends only on the parameter measuring the apparent parallelism and the average vector length of the algorithm. The performance of a family of methods for solving Poisson's equation was optimized on the basis of the above characterization.

The family of methods used, known as FACR(L) algorithms, involves the optimum combination of Fourier analysis in the x-direction and block cyclic reduction by lines in the y-direction. L is the number of stages of line-cyclic reduction that are performed before Fourier analysis takes place.

The Delft Parallel Processor Toward a DMIMD Processor

Professor L. Dekker reviewed the progress of the work at Delft on parallel processors during 1981-84. He stated that a parallel processor at Delft has been operational since mid-1981. The system completed in 1981 was a small MIMD processor consisting of one processing module with eight time-parallel subprocessors called processing elements. The processing module consists of processing elements for arithmetic/Boolean processing; an on-line intelligent display for real-time observation during a parallel run; a trajectory memory for storing time trajectories; and an off-line display for observation from the trajectory memory after a parallel run.

The 1984 system is also a MIMD processor consisting of one processing module with a maximum of 16 arithmetic/Boolean processing elements. A processing module for the 1984 system consists of the following:

- Processing elements for arithmetic/Boolean processing, state observation processing, and trajectory observation processing.
- An off/on-line user interaction processor.
- An on-line intelligent display for real-time state observation during a parallel run.
- An off-line intelligent display for trajectory observation after a parallel run.

The 1984 parallel processor can functionally be subdivided into: (1) the modeling and observation processing section, and (2) the experimentation processing section.

A processing element for the 1984 system consists of:

- An input buffer, allowing the sampling at each data transfer time of 24 input variables in one cycle or 48 input variables in two cycles.
- A variable memory of up to 64K words of 32 bits.

- A program memory with paging of up to 64K words of 16 bits.
- Two arithmetic processors.
- A Boolean processor.

Pyramidal Architecture

Professor V. Cantoni spoke on "Pyramidal Architecture for Image Processing." In the past, researchers have explored two main approaches to the fast processing of images, particularly for low-level applications: the array of processors and the pipeline of processors. Recently a new proposal for merging the advantages of both approaches has been suggested by a few scientists and is generally known as the hierarchical computation structure. Alternative names for these architectures are: (1) "perpection cones," introduced by L. Uhr, who is particularly interested in developing models of perception; and (2) "pyramids," introduced by C.R. Dyer and others interested in recursive data structures (quadtree, octree, pyramid).

A pyramid structure also may be considered as the three-dimensional extension of the two-dimensional binary tree. Three main features of this structure suggest its usefulness in image processing; the first is related to the well-known order logarithmic dependence for the interprocessor communication supported by this topology; the second exploits interplane communication for the implementation of a "planning" strategy using images at different resolution levels; the third depends on the possibility of different images flowing toward the apex in a pipeline mode.

To fully exploit these advantages, Cantoni and his colleagues have designed a pyramidal architecture for parallel image analysis (PAPIA). It is a multiprocessor pyramid architecture made of tapered layers of processors, each layer being a truly SIMD machine. Moreover, different layers may execute different instructions, thus PAPIA becomes a multi-SIMD processing system; alternatively, a given subset of layers may operate in the SIMD mode.

Cantoni discussed the main features of the new system: the image distribution on the processors, the processor-interconnection scheme, the bit serial arithmetic, the global features of the system, and the input/output modality.

Discussion

During the discussion period there were no long debates. The discussion consisted mainly of questions (Q), relatively brief answers (A), and comments (C).

Q. Has the correction of multiply on the CRAY machine been analyzed?

A. (Wheeler) It is too small to matter. You can call the CRAY machine a 62-bit rather than a 64-bit machine.

Q. What is the best way to compare the performance of machines--operations per cycle, millions of instructions per second, or with bench-mark problems?

A. (Granlund) An unambiguous comparison is very difficult to make.

C. (Wheeler) One should consider how difficult the machine is to use, how difficult to program, and its cost.

C. (Giloi) Some floating point machines are very powerful but very hard to program.

C. (Dekker) One takes into account speed and usability.

C. (Wheeler) Compile time is a very important element.

C. (Dekker) What about the distribution of the compilation function?

C. (Anceau) Speech recognition, for example, needs a special-purpose computer. Why share a large computer?

C. (Granlund) Specialization is very important. Either do purely serial processing well or specialize for the particular task.

Q. (Rawlings) Please comment on the question of fault tolerance and reliability.

A. (Giloi) Multiprocessing is an asynchronous process. One cannot analyze a fault from the outside. Constant self-diagnosis in each processor is

necessary. We cannot determine the global state as in the Von Neumann machine.

C. (Dekker) Parallel processors can use idle time for self-testing.

C. (Evans) I'd call it background reliability testing.

C. (Cordonnier) One must add extra control procedures.

C. (Anceau) Large VLSI systems are very reliable. Faults usually occur at the pins--not in the integrated circuits. One can use duplication of circuitry on the chip.

C. (Wheeler) That checks the chip but not the system.

C. (Giloi) The problem is like that of software--very difficult to diagnose.

C. (Wheeler) Synchronized arbitration is a choice at a fixed time. For fault tracing I'd recommend use of a real user program rather than a special program.

C. (Granlund) We can't always detect faults, but we can disconnect one pipeline, repeatedly, and thus isolate the problem to a particular pipeline.

C. (Schneck) In a NASA launch there was an error in the *checking* circuitry.

Q. (Evans) We've talked about machines that go back 20 to 30 years. There were other than Von Neumann machines then. What will be the survivors of today's machines?

C. (Hockney) ENIAC was a parallel machine having 20 adders. It was abandoned because of difficulty of programming it.

Q. (Evans) What about the survivability of vector machines?

A. (Dekker) In practice a vector machine is a special-purpose machine. However, the distinction between general-purpose and special-purpose machines is diffuse because you can only deal with part of the real problem. In the past, one used only general-purpose sequential processors or designed a special-purpose machine. Special-purpose machines have always been for special jobs. Now we tailor-make systems. Will we have tailor engineers designing systems to the needs of the users?

C. (Schneck) Some people ignore the vector capability in the CRAY machine and use it as a fast scalar machine.

C. (Wheeler) For special-purpose machines, the price depends on the number bought.

C. (Giloi) The software crisis is not due to the sequential nature of the Von Neumann machine but to semantic gaps, lack of protection of data objects, etc.

C. (Wheeler) All current Von Neumann machines have some protection. Part of the software crisis is due to the fact that nobody can write 100 lines of code without errors.

C. (Giloi) The architecture of machines should support protection and the semantic gap. A hierarchy of interpretive levels of protection should be provided by firmware. Alternatively a hierarchy of machines should provide this protection. A top-down design has goals given by application, language, reliability, performance, and fault tolerance.

C. (Wheeler) Languages and suggestions come along to put constraints and condition to prevent errors. However, instruction can still be modified.

C. (Schneck) For a program with several users one must preserve the status of a program so that each user will know the situation.

C. (Wheeler) There is no hierarchical restriction to the propagation of errors. Therefore, you have to work with a system with some errors.

C. (Schneck) The procedure "return to launch site" in NASA systems cannot be tested.

C. (Wheeler) At Harvard printed tables were carefully checked with guaranteed machines. But there were printing errors due to the inking, etc.

C. (Gurd) Proof of security simply says there is no leak beyond a certain tolerance.

C. (Cordonnier) I have designed too many computers. We can invent many new machines, but I think we have very few new ideas to apply to computer

architecture. Now we need to get new ideas about human behavior to use as a model for computer architecture. We need to investigate new possibilities for algorithms.

C. (Wheeler) There is room for experimentation. It takes 5 to 10 years to put experimental effort into productive results. It takes a year or two to train the users. We can produce many examples of parallel machines.

C. (Gurd) We have spent 8 years on the data-flow computer. We thought it would be a good vector processor. This proved to be wrong. It can be used to boost irregular parallel problems. We break the problems down into subproblems and optimize solutions to the subproblems.

C. (Dekker) I believe in a step-by-step approach to design which must incorporate possibilities that are now available. You need to be sure you can change your strategy from step to step. We should get together, pool our ideas, and see where we can go from there. I believe that to have impact over the next years we must have more intense contact.

C. (Granlund) That's a good idea when you get to the point of standardizing, but before that it may be counterproductive.

C. (Rawlings) What about synergism through the use of ARPANET?

C. (Evans) The question is when to stop simulation and to start building.

C. (Gurd) There is a point where simulation is no longer helpful.

Conclusion

From the presentations and discussion it is possible to draw a number of conclusions. As evidenced by Paul Schneck's comments, more computing power is needed due to the changing role of computers in basic scientific research. Simulation has been used instead of experimentation since early in the computer era. However, the use of simulation has expanded greatly in aircraft design, statistical mechanics, astrophysics, and nuclear physics. The report of the panel on "Large Scale Computing in Science and Engineering," under the sponsorship of the US Department of Defense and the National Science Foundation, lists the following areas in which dedicated supercomputers are in use: nuclear weapons research, atmospheric sciences, magnetic-fusion energy research, aeronautical research and development, nuclear-reactor theory and design, petroleum engineering, geophysics, and intelligence. Other areas listed as having need for supercomputers are: computational physics, computational mechanics and structural design, ocean sciences and underwater acoustics, computational chemistry and chemical engineering, VLSI and circuit design, nonlinear optics and electromagnetic theory, and computational fluid dynamics.

To meet this need for greater computing power, it is necessary to move from the single processor system to parallel processing. The limit for a single processor appears to be somewhere near 10^9 floating point instructions per second, and present technology is nearing that limit. Parallelism in design appears to be the alternative to follow, and much of the work described in this workshop was in this area. Architectural alternatives in parallelism ranged over SIMD and MIMD systems, a data-flow system, and a graph reduction system. Applications envisioned ranged from high-speed scientific computing to picture processing and robot control.

It is clear that university research and development in computer architecture in the seven countries represented at the workshop is on the forefront and is proceeding well. Part of the motivation may have been generated by the Japanese plan for a fifth generation of computers. However, a more important and more basic consideration is the clearly seen need for more powerful systems to solve the problems of the present and the future.

APPENDIX A: WORKSHOP PARTICIPANTS

Professor Francois Anceau
IMAG
B.P. 53, 38041 Grenoble Cedex
France

Professor Virginio Cantoni
Systems & Computer Science Department
Strada Nouva 106/C
University of Pavia
27100 Pavia
Italy

Professor Vincent Cordonnier
Université des Sciences
et Techniques
Computer Science Department
59655 Villeneuve D'Ascq Cedex
France

Professor L. Dekker
Department of Mathematics
and Informatics
Delft Institute of Technology
134 Julianalaan
P.O.B. 356, 2600 AJ Delft
The Netherlands

Professor D.J. Evans
Department of Computer Science
Loughborough University of Technology
Loughborough, Leicester
United Kingdom

Mr. Anthony Field
Department of Computer Science
Imperial College of Science and
Technology
South Kensington
London SW7 2AZ

Professor Wolfgang K. Giloi
Technische Universität Berlin
1000 Berlin 12
Strasse Des 17 Juni 135
Berlin
Federal Republic of Germany

Professor G.H. Granlund
Faculty of Technology
Linköping University
581 83 Linköping
Sweden

DR. John Gurd
Victoria University
of Manchester
Oxford Road
Manchester M13 9PL
United Kingdom

Professor R. Hockney
Computer Science Department
University of Reading
Reading RG6 2AH
Berkshire
United Kingdom

Professor U. Schendel
Freie Universität Berlin
Institut für Mathemik
III (WE3)
Arnimallee 2-6
1000 Berlin 33 (Dahlem)
Federal Republic of Germany

Professor D.J. Wheeler
Computer Laboratory
Cambridge University
Cambridge CB2 3QG
United Kingdom

US Government Attendees

Dr. J.F. Blackburn
US Office of Naval Research
London

Dr. James W. Daniel
US Office of Naval Research
London

Mr. Robert Carpenter
US Department of Commerce
Washington DC

Major J.B. Rawlings
US Air Force
London

Dr. Paul Schneck
US Office of Naval Research
Arlington

Dr. David Squire
US Army
London